University of Waterloo
Faculty of Engineering
Department of Electrical and Computer Engineering

Final Report

Canary: A Personal Air Quality Monitor

Group Number: 2025.33

Group Members: Taylor McNabb (20886459)
Ken Jiang (20877405)
Prabhjyot Singh (20891880)
Martin Kusnir (20897496)

Advisor: William Bishop

February 24, 2025

Abstract

Globally, air pollution is the fourth leading risk factor for early death, responsible for nearly 7 million deaths annually. Unlike other risk factors such as diet and tobacco use, people are generally ill-informed about the quality of the air they breathe. This project addresses this issue by designing a portable, battery-operated device that empowers people to know and act on the air quality of their surroundings. For ease of use, the device can operate standalone, with a low-power display and buttons to display readings, navigate menus, and change power profiles. The device can also pair with a smartphone over Bluetooth for richer data visualization and advanced customization. The device includes sensors to detect carbon dioxide, particulate matter, and more, all mounted onto a custom PCB. Customizable alerts inform users of high pollutant levels tailored to their risk. Overall, the design combines knowledge of wireless communication, circuit design, firmware, and software. The advantages of this design over alternatives are its ability to function entirely standalone and the range of included sensors. Compared to other products on the market, Canary sets a new benchmark for portability, battery life, and data quality.

Acknowledgments

We would like to thank our advisor, Dr. William Bishop, for his unrelenting support and advice throughout our project.

Contents

List of Figures

List of Tables

1 High Level Description

1.1 Motivation

Air pollution is a critical public health issue, responsible for 7 million deaths each year and ranking as the fourth leading risk factor for early death worldwide [1]. Over 99% of the world's population is exposed to unhealthy air, increasing the risk of respiratory and cardiovascular diseases, stroke, cancers, and more [1].

Although public monitoring of air quality is improving [2], it is estimated that nearly half of air pollution related deaths are caused by indoor air pollution [3]. Pollutants such as particulate matter ($PM_{2.5}$) and volatile organic compounds (VOCs) can accumulate in homes, workplaces, and other environments, posing a severe risk to human health [4].

1.2 Project Objective

The objective of this project is to design a portable air quality monitoring device and an associated mobile application to allow people to monitor and act upon the air quality of their environment.

1.3 Block Diagram

The high-level design of the proposed device and its associated application is described in Figure 1 on the following page. Components and subsystems are color-coded to indicate the type of design required. To reduce clutter in the diagram, connections between the power subsystem and the sensors have been omitted.

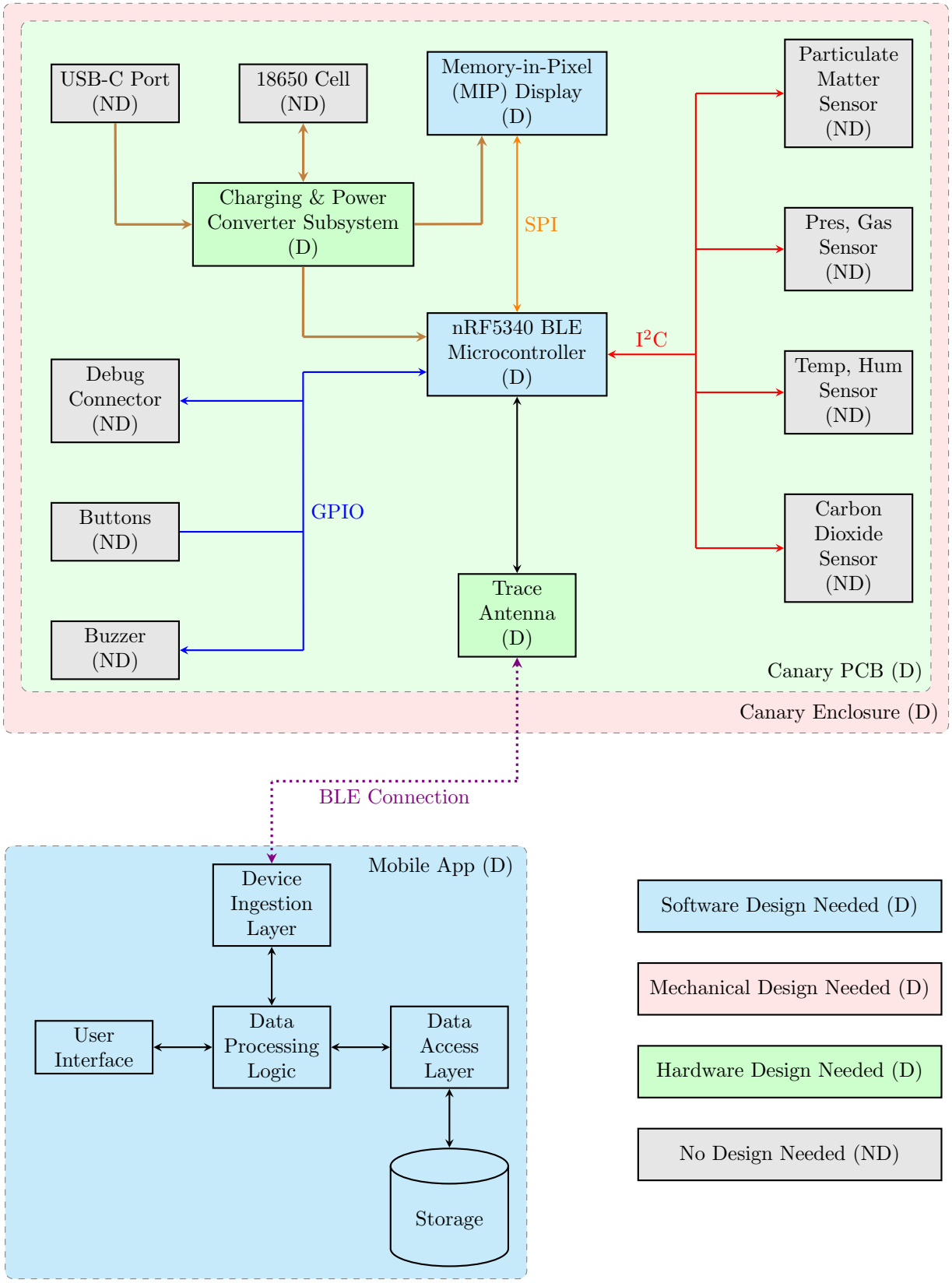


Figure 1: High-level Block Diagram

2 Project Specifications

The project specifications are classified as either functional or non-functional. Furthermore, within each category, the specifications are deemed essential or non-essential based on their importance to achieving the project objective. All specifications are numbered and associated with a specific subsystem as described in Figure 1.

2.1 Functional Specifications

Eight essential functional specifications were identified.

#	Subsystem	Specification	Description
1	Sensors	CO_2 Measurements	The CO_2 sensor has a baseline accuracy of at least $\pm$100ppm between 400 and 5000 ppm.
2	Sensors	Particulate Matter Measurements	The particulate matter sensor can bin particles into at least two categories ($PM_{2.5}$ and PM_{10}) and has a baseline accuracy of at least $\pm 75\,\mu g/m^3$.
3	Power	Battery Life	The device is capable of at least 24 hours of continuous operation in its default mode.
4	Power	Charging	The device can be recharged via USB-C connection in less than five hours.
5	MCU	Data Storage	The device can locally store at least 24 hours of recorded data.
6	MCU	BLE	The device can pair with a phone and send data from a distance of at least 3 metres.
7	App	Power Profiles	The app can be used to configure the sampling frequency of the sensors on the device.
8	App	Historical Data	The app can store and display historical data.

Table 1: Essential Functional Specifications

Five non-essential functional specifications were identified.

#	Subsystem	Specification	Description
9	Sensors	Other Measurements	The device can measure parameters such as temperature, humidity, and pressure.
10	Power	Replaceable Battery	The device battery is user-replaceable without any tools.
11	Enclosure	Mounting	The device can be mounted using a conventional system such as a 1/4"-20 thread, GoPro finger mount, or similar.
12	App	Live Dashboard	The application can display live (previous measurement or 60s, whichever is greater) data when the device is connected over BLE.
13	App	Data Export	The app can export data to a common file format such as `.csv`.

Table 2: Non-Essential Functional Specifications

2.2 Non-Functional Specifications

Five essential, non-functional specifications were identified.

#	Subsystem	Specification	Description
14	Sensors	Lifespan	The sensors have a mean time between failures (MTBF) of at least five years.
15	PCB	Antenna	The PCB has a trace antenna to allow the MCU to communicate over Bluetooth.
16	Enclosure	Weight	The device weighs less than 1 lb (454 g).
17	Enclosure	Dimensions	The device is no larger than $14 \times 14 \times 7$ cm.
18	App	Accessibility	The application supports OS-provided accessibility features such as screen readers.

Table 3: Essential Non-Functional Specifications

Two non-essential, non-functional specifications were identified.

#	Subsystem	Specification	Description
19	App	Set up time	The application takes less than two minutes to connect to a device for the first time and receive data.
20	Device	Cost	The components in the prototype cost less than $400.

Table 4: Non-Essential Non-Functional Specifications

3 Detailed Design

3.1 MCU

The Nordic nRF5340 microcontroller (MCU) was chosen for its high performance, low power consumption, and familiar toolchain. The custom firmware is based on the Nordic nRF Connect SDK, which itself is a fork of the Zephyr Real-time Operating System (RTOS). This section discusses the firmware, peripherals, and connectivity of the MCU. The integration of the MCU into the overall PCB is discussed in section 3.4.

Sensor Ingestion

There are four environmental sensors in Canary, all of which are accessible to the MCU over the I^2C interface. Three of the four sensors lack proper "in-tree" (native) support in Zephyr, so custom drivers were written to adapt the manufacturers' driver code to work with native Zephyr calls such as `sensor_sample_fetch()` and `sensor_channel_get()`. This simplifies the firmware by abstracting away the intricacies of each sensor's communication channels.

User Interface

Canary has an on-board user interface, consisting of a directional joystick for inputs and both a display and buzzer for output.

A 5-way joystick (Alps Alpine SKRHAAE010) was used for user input. This joystick was chosen for its compact size, reliability, and integration of 5 directions (4 cardinal directions and center press) into a single device. This choice greatly simplifies the enclosure design, which is further discussed in section 3.5.

The display shows real-time sensor readings and indicates the state of the device. Although an e-ink display was used in initial prototypes, the final design uses a 2.7" Sharp Memory-in-Pixel (MIP) display, model LS027B7DH01. This display offers a higher resolution of 400 x 240 pixels, and has much better refresh rates than an e-ink display. Although the power consumption is slightly higher than e-ink, it is still very low at around 175 µW with a 1Hz update rate.

Sound output occurs through a piezoelectric sounder (a buzzer without a built-in driver circuit). The sounder is driven with an external FET connected to one of the MCU's GPIO pins. By varying the frequency of the input square wave, the tone of the sounder can be changed. Our design uses lower frequencies to indicate user input and higher frequencies for alerts.

Storage

The nRF5340 has 1MB of built-in flash, which is sufficient to store the firmware image and device settings (sensor polling interval, alert configuration, etc.). To enable longer-term storage of recorded values, Canary has an additional 32 MiB QSPI NOR flash. Although microSD card support was considered, it was not included in the final design since there is no reason for a user to remove the card (all data can be exported through the app).

The Zephyr RTOS exposes a persistent storage API called Non-Volatile Storage (NVS), which is used to store previous sensor readings. NVS is a key-value storage with history, allowing previous sensor values to be read with `nvs_read_hist()`. A single key is used for each type of measurement, with the data containing both the value and a timestamp (since different sensors can have different polling rates).

There is also a limited need to store settings, such as Bluetooth connection information and alert configuration. Zephyr exposes a settings API, which stores settings keys and values directly on the nRF5340's flash. These values can be read concurrently during runtime, and updates are automatically committed to the flash storage.

Communication

Bluetooth Low Energy (BLE) communication is used to pass data between Canary and its companion application. The nRF5340 is a dual core MCU, with an "application" core and a "network" core. The firmware design consists of two images, a generic BLE host controller interface (HCI) image for the network core and our custom firmware on the application core. Inter-process communication (IPC) through a shared region of memory is used to pass messages between the cores. For example, when the application core receives a new sensor value, it stores the intended BLE communication in the shared memory and the network core ingests this and handles the actual transmission. Although this dual-core configuration is not required, testing determined that running the BLE stack on the application core directly could lead to poor display performance.

Regarding the BLE communication itself, sensor data is transferred from Canary to the companion application using the BLE Environmental Sensing Service, which is part of the BLE specification. Within this service, several characteristics are exposed for each of the sensors (temperature, humidity, CO_2 concentration, etc.) and Canary notifies connected devices when a new value is available for any of the characteristics. When the companion application needs to communicate with Canary (e.g. to transfer settings), the BLE Object Transfer Service (OTS) is used, since the data format is unique to Canary.

3.2 Sensors

The following sensors have been selected for the design. The rationale for choosing each sensor is explained below, in addition to discussion surrounding their power consumption. In particular, we have carefully selected sensors which can operate continuously in a wide range of conditions without data quality concerns.

Sensor	Measurement Type
Sensirion SCD30	CO_2
Sensirion SPS30	Particulate Matter
Sensirion SHT45	Temperature, Humidity
Bosch Sensortec BME688	Atmospheric Pressure, Gas (VOC)

Table 5: Final Sensor Selection

3.2.1 Sensor Selection

CO_2 Sensor

The CO_2 sensor is one of the most important sensors in our design, since CO_2 has a significant impact on indoor air quality and a typical person is typically unable to measure or filter it. With this in mind, specification #1 sought a baseline accuracy of ±100ppm between 400 and 5000 ppm. Although several sensors on the market can achieve this accuracy, many are susceptible to *drift*, a gradual shift in their reported values.

There are three common CO_2 sensor types: Non-Dispersive Infrared (NDIR), Electrochemical, and Metal Oxide Semiconductor (MOX). Of these three, NDIR sensors offer the best accuracy and lifespan, and they are not susceptible to cross-sensitivity biases (i.e. they do not confuse other pollutants with CO_2). However, NDIR sensors are still susceptible to long-term sensor drift and temperature/humidity induced variances.

To address the latter issue, many NDIR sensors have integrated temperature and humidity sensors to reduce variance caused by transient air conditions. However, long-term sensor drift remains an issue due to slow degradation of the infrared LED. The typical method for combating this drift is automatic self-calibration (ASC), which identifies the lowest reading over a period of time and assigns it to 400ppm, the baseline outdoor CO_2 concentration. There are several NDIR sensors which offer high accuracy through ASC algorithms, such as the Senseair S8 and Sensirion SCD40/41.

However, relying on ASC is unacceptable in some conditions. We intend to permit users to measure air quality in all conditions, including poorly-ventilated indoor spaces. ASC

algorithms fail in these cases because they are never exposed to sufficiently fresh air to establish a proper baseline. For example, if the CO_2 levels in a home never fell below 1000ppm, the ASC algorithm would slowly introduce drift until it reported that the air inside was "fresh" (i.e. around 400ppm of CO_2).

To address this, we have chosen the Sensirion SCD30 dual-channel NDIR CO_2 sensor. The SCD30 has an exceptional accuracy of ±30ppm between 400 and 10000 ppm, exceeding the requirements in specification #1. More importantly, it features two separate measurement channels, with one channel exposed to the outside air and one sealed with a reference gas. This configuration allows for automatic calibration to occur without a reference CO_2 concentration, allowing the SCD30 to operate reliably for long periods of time without the use of ASC algorithms.

Particulate Matter Sensor

Specification #2 requires the Particulate Matter (PM) sensor to bin particles into $PM_{2.5}$ and PM_{10} and maintain an accuracy of $\pm 75\,\mu g/m^3$. We selected the Sensirion SPS30, which beats the specification with binning into four categories (PM_1, $PM_{2.5}$, PM_4, and PM_{10}) and a baseline mass concentration precision of $\pm 25\,\mu g/m^3$. Additionally, the SPS30 features enhanced precision for $PM_{2.5}$ measurements, with an accuracy of $\pm 5\,\mu g/m^3$ + 5% m.v., ensuring highly precise measurements of the particles most harmful to human health. This sensor also has above-average physical durability and lifespan, which is important in our portable form factor.

Compared to alternatives such as the Plantower PMS5003/7003 and the Sensirion SEN5X series, the SPS30 has a better form factor and slightly lower average power consumption. Furthermore, the SPS30's performance has been validated with independent laboratory testing, which is not the case for similar PM sensors [5].

Temperature & Humidity Sensor

Specification #9 indicates that the device should be capable of measuring temperature, humidity, and pressure. There is no mention of accuracy since these measurements are not critical to assessing air quality, however, we would like to strive for the best option with reasonable power consumption and packaging.

With this in mind, we have selected the Sensirion SHT45. It is exceptionally accurate, with typical accuracy of ±1.0% R.H. and ±0.1°C. As is the case with all polymer-based capacitive humidity sensors, the SHT45 can develop a reversible humidity offset when exposed to prolonged periods of high humidity (> 90% R.H.), which is known as *creep*. However, this particular sensor has an integrated heater which can be cycled during high-humidity periods to dry out the sensor and mitigate creep. The use of this heater will ensure lasting data accuracy in all conditions, which is one of our design goals with Canary.

Gas Sensor

The selection of gas sensors poses a challenge due to the variety of products on the market and the challenges in providing meaningful quantification of the risk associated with Volatile Organic Compounds (VOCs). There are three primary types of VOC sensors: photoionization detector (PID), flame ionization detector (FID), and metal oxide semiconductor (MOX). Although PID and FID sensors offer the highest accuracy, they typically target detection of individual compounds, and neither can be integrated into our desired design due to their large size and high power consumption. Furthermore, the most important VOC metric for indoor air quality is the total quantity of VOC (tVOC), however, no individual sensor can measure this metric directly.

Consequently, we have selected the Bosch Sensortec BME688, a MOX sensor that offers a normalized Indoor Air Quality (IAQ) measurement. With this sensor choice, our aim is to help users quantify the risk associated with a variety of VOCs instead of providing precise measurements of certain pollutants such as ethanol or formaldehyde.

It should be noted that the BME688 has an integrated temperature, humidity, and pressure sensor. However, the gas-sensing operation involves heating a MOX plate to upwards of 200°C, which introduces significant discrepancy in the reported values for other measurements. Therefore, in order to use the gas sensing capabilities of BME688, we need to outsource the temperature and humidity measurements to another sensor.

3.2.2 Power Consumption and Polling Rate

Managing power consumption is critical in a battery-powered design. With the sensor subsystem, power consumption is a function of the sensor polling rate. Selection of a suitable polling rate requires a model of the corresponding current draw, which we will describe with the following formula:

$$I_{\text{avg}}(T_{\text{poll}}) = \frac{T_{\text{meas}} \cdot (I_{\text{meas}} - I_{\text{idle}})}{T_{\text{poll}}} + I_{\text{idle}}$$

where T_{poll} is the sensor polling interval (time between measurements).

The following parameters are defined for each of the sensors:

- V_{DD}: Supply voltage
- I_{idle}: Idle current (sensor on, I^2C enabled, ready to measure but not measuring)
- I_{meas}: Average current in measurement mode
- T_{meas}: Time in measurement mode needed to acquire a single measurement

For many of the sensors, these values are readily available in their respective data sheets. However, the Sensirion SCD30 current consumption is more complex to model since there are three primary power-consuming components: the IR emitter, the onboard microcontroller, and the photodetectors. The manufacturer does not specify the duty cycle or current consumption of these three components, however, the following known values are provided:

Polling Interval (s)	Average Current (mA)
2	19
15	6.5
30	5.6

Table 6: SCD30 Average Current at Various Polling Intervals

The manufacturer specifies that the maximum sampling frequency is 1 Hz ($T_{meas} = 1$ s). After applying a regression on this data substituted into our earlier average current function:

$$I_{\text{avg}}(T_{\text{poll}}) = \frac{1 \cdot (I_{\text{meas}} - I_{\text{idle}})}{T_{\text{poll}}} + I_{\text{idle}}$$

we get an R^2 of 1 with $I_{\text{meas}} = 33.38$ and $I_{\text{idle}} = 4.61$.

Finally, we can create the following table:

Parameter	SHT45	BME688	SCD30	SPS30
V_{DD} (V)	3.3	3.3	3.3	5.0
I_{idle} (uA)	0.08	0.29	4610	330
I_{meas} (mA)	0.320	0.9	33.4	55
T_{meas} (s)	0.0069	1.4	1	60

Table 7: Power Consumption Parameters of Selected Sensors

From this table, we can conclude that the power consumption of the SHT45 and BME688 is not a significant concern within the design. However, SCD30 and SPS30 have comparably high current requirements.

Figure 2 below depicts the average current draw of each sensor as a function of the polling interval. Note that the domain of each plotted curve begins at the minimum sampling interval (as lower values are not achievable).

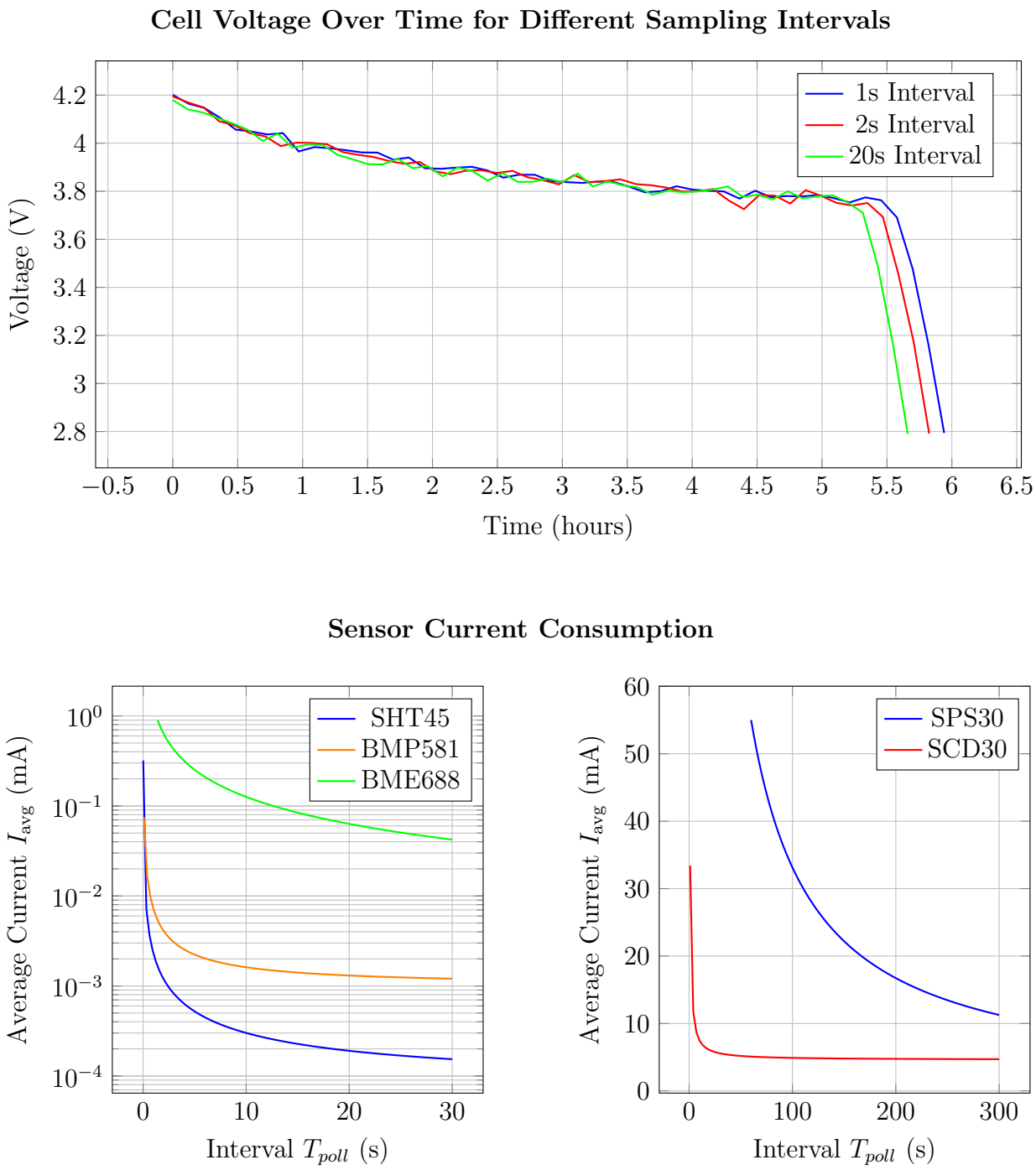


Figure 2: Average Current vs Polling Interval for Selected Sensors

From these graphs, it is evident that adjusting the polling interval between reasonable values (> 5 s) does not have a significant impact on the current draw of the SHT45 or BMP581. The BME688 shows a more pronounced difference, however, even at the highest polling rate, the current draw is sufficiently low.

In contrast, reducing the polling rate of the SCD30 has a very significant impact on power consumption. The current draw can be reduced from over 30 mA to around 7.5 mA by decreasing the polling rate from once per second to once every ten seconds. This is still a very acceptable frequency for CO_2 measurements, as they are unlikely to fluctuate more quickly.

Unfortunately, the SPS30 consumes high power even at extended polling intervals due to the long measurement duration (60 s). However, significant improvement is still possible at extended polling intervals. With a polling interval of 5 minutes, the average current consumption is just 11 mA. Depending on the application, this may still be an acceptable interval. For example, if outdoor air quality is being monitored, it is unlikely that the data will fluctuate significantly in five minutes. However, with indoor pollutant levels, this polling interval may lead to inconsistent data. User configuration will allow for the best mix of battery life and performance to be selected for a given application.

3.3 Power

Canary is powered by a single 18650 (lithium-ion) cell due to their power density, reliability and abundance. Most high performance cells typically accept a maximum charge current of 4A. To meet specification #3 (24 hour battery life) and #4 (USB-C charging in under 5 hours) we require power ingress exceeding the capabilities of the USB 2.0 protocol. Accordingly, we have the following options:

- USB 3.0/USB 3.1: 5V * 900mA = 4.5W
- USB BC 1.2: 5V * 1.5A = 7.5W
- USB Type-C 1.2: 5V * 3A = 15W
- USB Power Delivery 3.0: 20V * 5A = 100W

The maximum charge speed of the 18650 cell we plan to use is 16.8W. We chose the USB Type-C 1.2 specification (15W) since it can nearly saturate the maximum charge speed. In order to prevent damage to USB adapters which cannot supply 15W, a Channel Configuration (CC) Logic chip is required to identify the capabilities of connected chargers. For this role, we selected the NXP PTN5150 due to its low current consumption and I^2C interface.

The cell charger IC must satisfy the following requirements:

- Can charge our cell near its max capacity (4A)
- Supports I^2C interface for communication
- Power path management for operation without a battery
- Safety features (e.g. battery undervoltage protection, thermal regulation)

Based on these requirements, we have selected the Texas Instruments BQ25622E. Although the maximum charge current is only 3A, the features and form factor are exactly as desired and specification #4 can still easily be satisfied.

The remaining power topology is the selection of converters, which power our range of sensors and main MCU. We needed to consider three main factors when selecting these converters. Firstly, they must accept a wide input voltage range. Depending on the battery charge state, our system bus voltage can vary between 3.4V and 4.2V. The converters must function efficiently throughout this range. Furthermore, we will need both a 3.3V and 5V rail, so we will need regulators for both. Lastly, we need to consider the maximum current draw for each rail to size the regulators appropriately. Below is a table listing the components each rail provides and their associated maximum current draw.

Device	Voltage Rail (V)	Max Current (mA)
BME688	3.3	9.8
nRF5340	3.3	25
SHT45	3.3	100
SCD30	3.3	75
PTN5150	3.3	0.356
Status LEDs	3.3	70
SPS30	5	80
Buzzer	5	60
Screen	5	1

Table 8: Voltage and Maximum Current of Selected Components

From this we find the max current draw from each rail is as follows:

- 3.3V $\Rightarrow$ 0.280A
- 5V $\Rightarrow$ 0.141A

With these max current draws determined we select the following converters.

- 3.3V Rail: TPS6380 Buck-Boost Converter. Although our input should not drop below 3.4V, in order to maintain acceptable efficiency while the battery approaches 3.3V we opt to go with a buck-boost topology, which will accept a much wider range of input voltages and sustain an efficiency approximately 90% for the entire range.
- 5V Rail: TPS613222A Boost Converter. A very small footprint with minimal external components, which is still stated to reach above 90% efficiency for our input range.

With all of these considerations in mind, we end up with the following schematic for our power subsystem, shown below in Figure 3.

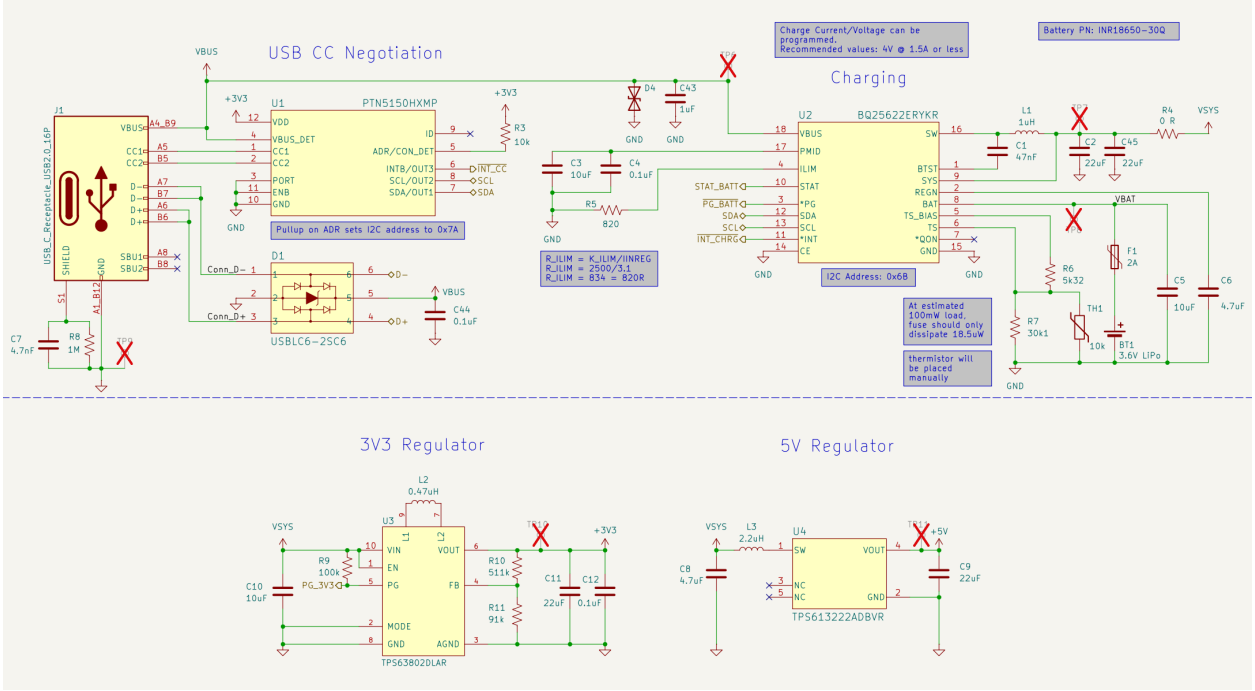


Figure 3: Schematic for Power Subsystem

3.4 PCB

The complexity of our project required us to implement a four layer PCB design, with the primary dimensions dictated by the 18650 cell and the SPS30 PM sensor. These components measure approximately 6.6 x 9.2 cm beside each other, which provided us with a good amount of space for the remaining components, with the SCD30 being the only large sensor left to place. Although a daughterboard was proposed, a slot in the main board was used instead to reduce the fabrication cost. This allows the SCD30 to sit flush with the PCB.

For the rest of the layout, priority was given to the location of the BLE antenna, USB-C connector, and display connector, as they could compromise the usability of the device. For the remaining sensors, temperature sensitivity was considered and mitigated in three ways. Firstly, the sensors are far from the battery and charging circuitry, as these are expected to generate the most heat. Furthermore, the enclosure features extensive passive ventilation to dissipate internal heat and give the sensors the best access to ambient air. Finally, the sensors have milled slots surrounding them to isolate heat transferring through the PCB.

Regarding routing, the I^2C bus was kept as short as possible (despite its seven endpoints), and the area above and below the BLE antenna was blocked to be free of any power or ground planes. The major challenge came from breaking out all of the nRF5340's pins, but after many iterations of swapping GPIO assignments, a suitable solution was found. Figures 4, 5, and 6 on the next page highlight the MCU layout density and overall design.

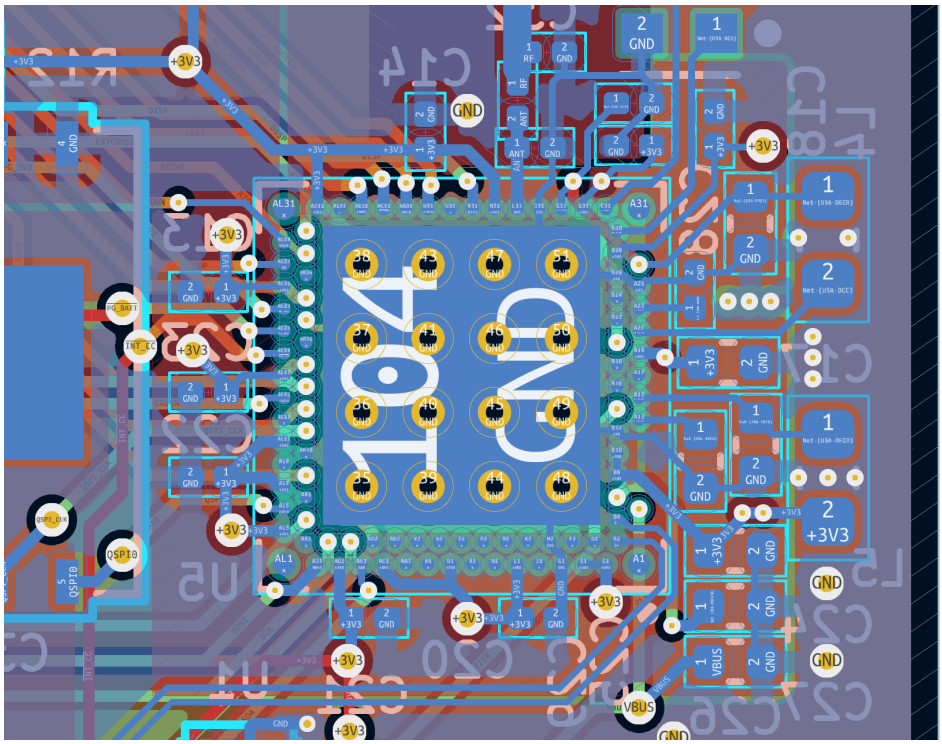


Figure 4: Breakout of the nRF5340

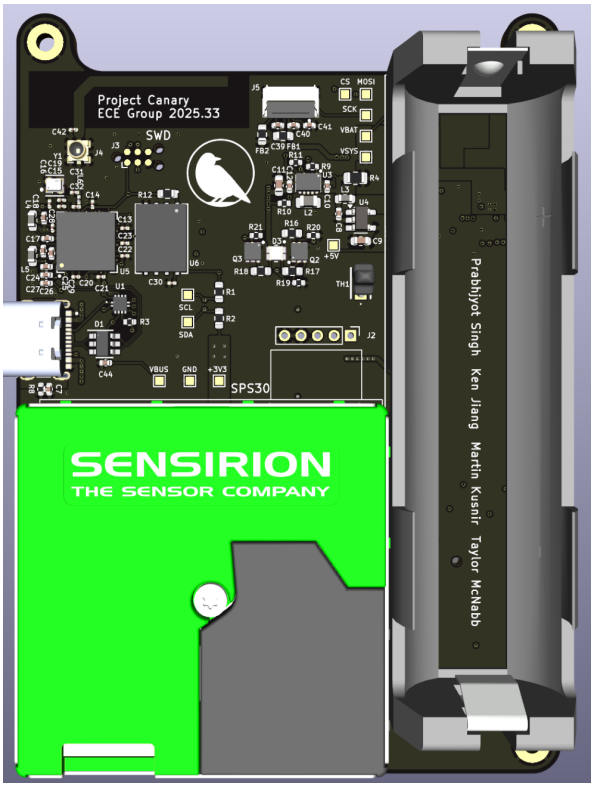


Figure 5: Front of board

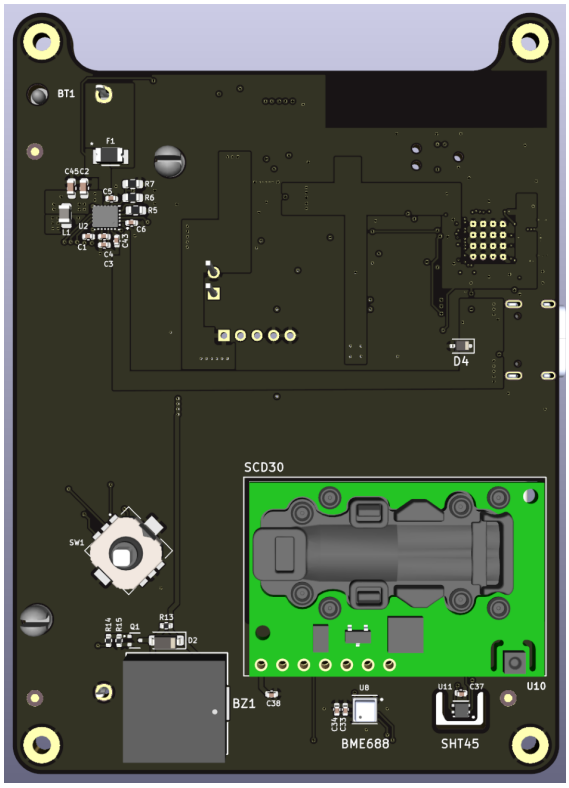


Figure 6: Rear of board

3.5 Enclosure

The enclosure is intended to protect the device, expose fresh air to its sensors, and permit easy mounting. It is constructed of both off-the-shelf and 3D printed components, and allows easy (tool-free) access to the battery and debug ports through removable, sliding doors in the back of the enclosure.

The 3D printed components are printed in Polylactic Acid (PLA) for its ease of printing and relative strength. Furthermore, its low melting point allows for easy installation of heat-set inserts, which are used in conjunction with Phillips head screws to allow the enclosure to be repeatedly assembled and disassembled. The design also features shims to ensure that the display and PM sensor cannot move within the enclosure, and 1/16" acrylic forms a protective cover over the display. There are several vents on the front and bottom of the case, providing ample fresh air for the sensors. Additionally, internal ducts ensure that the intake and exhaust for the PM sensor are properly separated.

The enclosure meets all of the original design specifications, measuring 97 × 71 × 28 mm and weighing 185 g with the battery installed. Figure 7 below depicts an exploded view of the entire assembly.

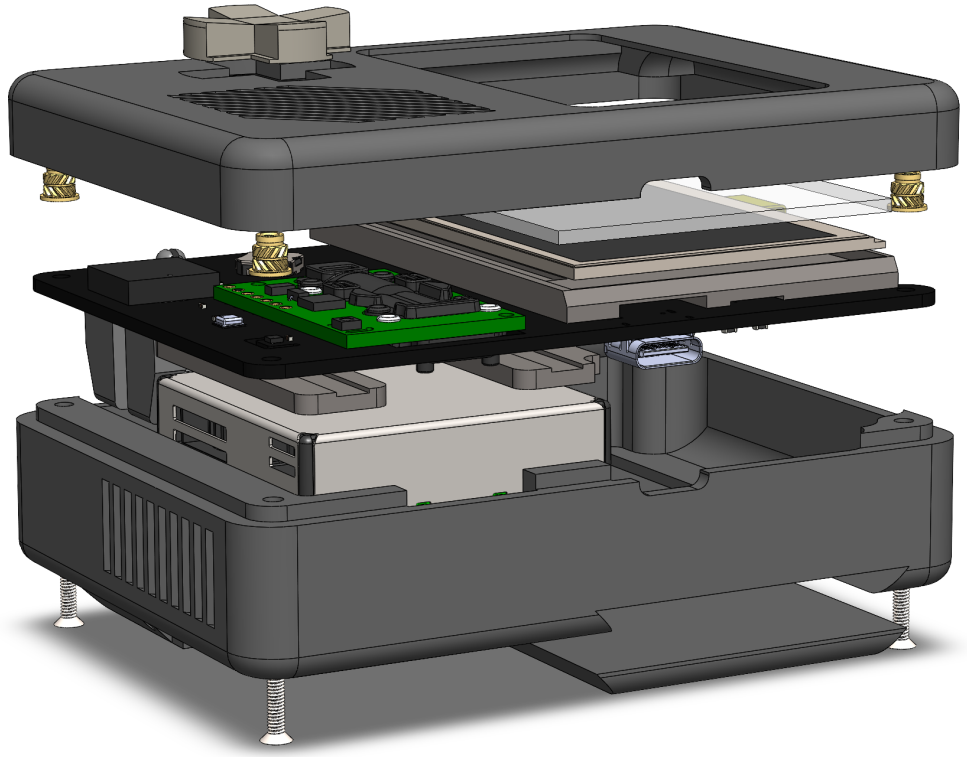


Figure 7: Exploded view of the overall assembly

3.6 App

The mobile application serves as a companion to the Canary hardware device. Due to the phone's larger screen size and more flexible control mechanisms (touchscreen, keyboard, etc.), features that would be unwieldy or impossible to implement on the hardware device are done so in the app.

Previously, the mobile application was developed using a platform called Expo, which allowed us to use React Native to write cross-platform applications in TypeScript. However, we found it difficult to interface with native components, especially Bluetooth. As Bluetooth connectivity was a key requirement for our app, we switched to native Android development using Kotlin. Although we are no longer able to leverage our existing background in TypeScript development, using native services like Bluetooth was made much easier, and we are still able to transfer our Java knowledge to Kotlin.

The UI for the app is designed with simplicity, accessibility, and ease of use in mind. Since the application is the easiest way to connect to the device to upload profiles and display data, the UI is designed to facilitate this process. The design process began with a rough hand-drawn version of the UI to ensure that all necessary components were included. The hand-drawn version was uploaded to Figma, where an initial wireframe was created, and the components were digitized. These designs were then iterated on until a satisfactory colour scheme and layout were achieved. The final appearance of the app is visible in section 4.3. Throughout this process, some features were simplified or removed to avoid unnecessary complexity.

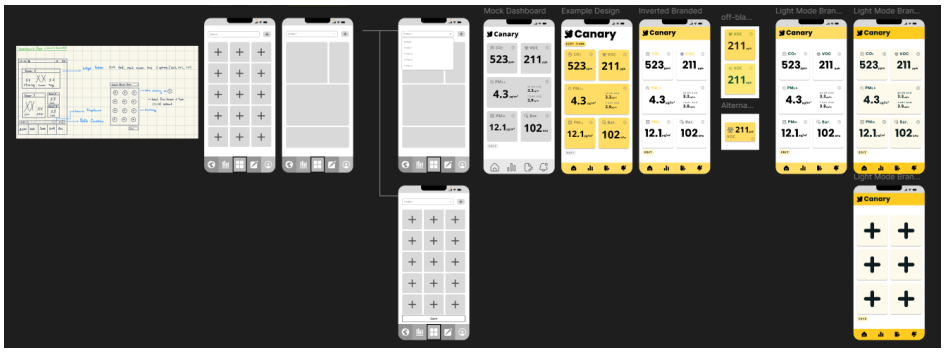


Figure 8: Design process for Dashboard Application Page

Figure 8 shows the design process from left to right for specification #12 (Live Dashboard). The design includes all components specified, such as the ability to display data for each sensor on the device. It also incorporates customization, allowing users to edit and select the tiles and sensor data relevant to them.

Figure 9 shows the design for the profile configuration and alerts page. These pages are designed to address Specification #7 (Power Profiles). The design process for this specifi-

Figure 9: Design process for Alert and Profile Management Application Page

cation is split into a profile page and an alerts page. The profile page allows the user to create separate profiles with different polling rates. The UI is designed to make it as easy as possible to add new sensors and save configurations on the phone. Once a change to a profile or a new profile is made, the "Push to Device" feature ensures that the user is aware of needing to push these profiles to the device. The alerts page is designed to be familiar, following the same design structure as the Apple alarm app.

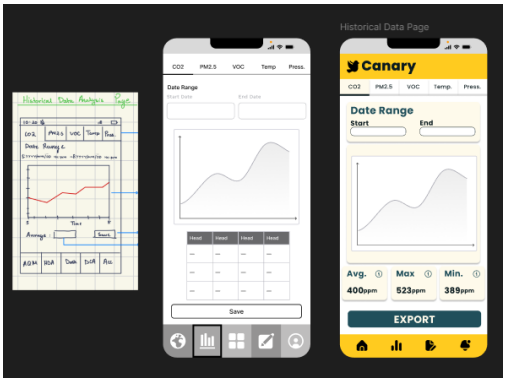


Figure 10: Design process for Historical Data Analysis Application Page

Lastly, Figure 10 describes the design process for the historical data page, which addresses specifications #8 and #13. By adding the export button, we enable the export of data for a desired time frame. Simple statistics such as the average, maximum, and minimum values provide key processed data. In all of the UI designs, we address both essential and non-essential specifications related to the mobile application. We discuss specification #18 (Accessibility) in section 4.3

3.6.1 Device Communication

One requirement for the mobile application is the ability to communicate bidirectionally with the hardware device. The app is able to do so using Bluetooth Low Energy (BLE) through the GATT protocol. The app allows the user to scan and connect to nearby Canary devices, initializing a link with the Canary. When characteristics from the environmental sensing characteristic are indicated, a callback function is invoked, parsing the bytes and

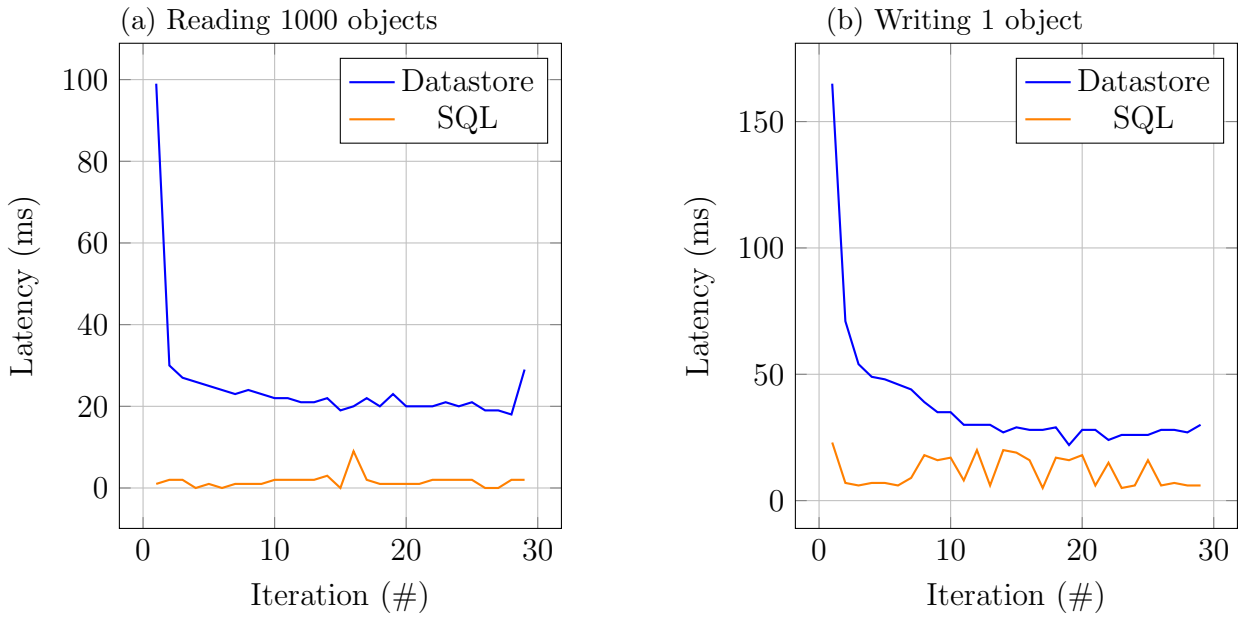


Figure 11: Latency vs Iteration for Preferences DataStore and SQL

saving them into the database. The BLE connection is persisted regardless of what page the app is on, and even when the app is in the background. Scanning for and connecting to a Canary takes mere seconds, meeting non-essential non-functional specification #19.

3.6.2 Data Storage

Data received from the device is stored locally on the phone in an SQLite database, fufilling essential functional specification #8. Our Android app uses the Jetpack library suite, allowing us to use the Room library to abstract many of the complexities of SQL. Data from the sensors are all stored in one table with the following schema:

```
CREATE TABLE IF NOT EXISTS sensor_data (id INTEGER PRIMARY KEY NOT NULL,
sensorName TEXT NOT NULL, value REAL, timestamp INTEGER);
```

Additionally, we use the Jetpack Preferences DataStore to persist smaller amounts of data that are less frequently accessed. This library allows us to store basic data types in a key-value store, such as sets, numbers, and strings. We leverage this library to store alert information and the layout of the dashboard.

To measure the performance of each data store, we perform repeated reads/writes (iterations) for common access patterns and measure the latency.

Analyzing historical data commonly involves reading many thousands of data points into the

past. According to Figure 8a, the SQL implementation of this commonly has a 20x lower latency, and does not face an initial latency penalty (as the Preferences DataStore must load the value from storage into RAM). Furthermore, looking at figure 8b, when writing 1 object like when new data is retrieved from a sensor, the SQL implementation again shows a large latency gap compared to Preferences DataStore. Due to the latency differences, the ease of querying, and the type safety guaranteed by the Room library, we use the SQL implementation for storing sensor data, but still use the Preferences DataStore for its simplicity for lower traffic data.

4 Prototype Data

4.1 Power Consumption

Upon receiving our prototypes from the manufacturer, we immediately noticed excessive power draw, both from the battery when operating on a cell and from the USB port when plugged in. In both cases, the average current draw was 500mA—an order of magnitude higher than anything on our board should consume. This issue was consistent across all five prototypes and remained unchanged regardless of the firmware flashed to the board. By inspecting the board with a thermal camera, we determined that the problem originates on the 3.3V rail but does not appear to be a full short, as the current draw is only a quarter of the 3.3V regulator's full capacity. After inspecting the board using X-ray imagery, we concluded that the excessive current draw is likely due to a manufacturing defect involving the μ-vias around the nRF5340, as shown in Figure 12.

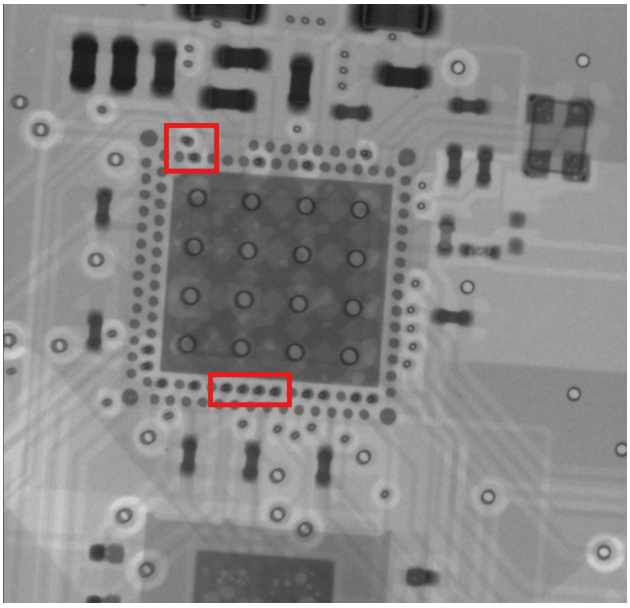


Figure 12: X-ray imagery highlighting the possible problematic μ-vias

As a result, our device's actual power consumption significantly exceeds its intended design, falling well short of the targeted 24-hour battery life and lasting only around 6 hours. However, we can still observe how different power profiles impact battery life. Figure 13 shows three plots of monitored cell voltage over time, each corresponding to a different SCD30 sampling interval. At its fastest setting, refreshing once per second, battery life is reduced to 5.6 hours. When the refresh rate is increased to once every 20 seconds, battery life extends to 6 hours.

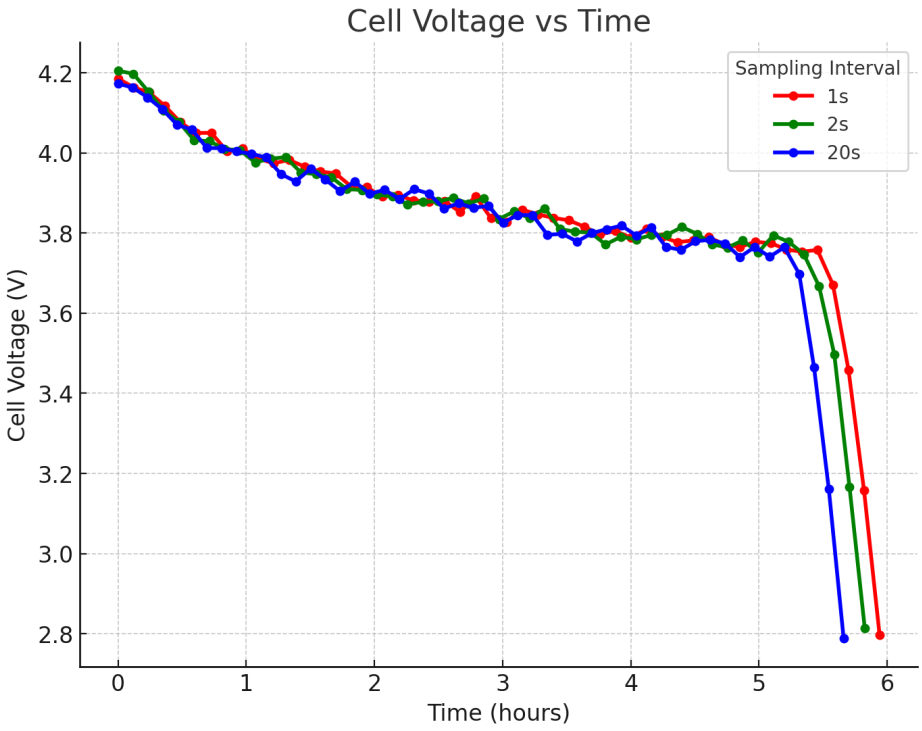


Figure 13: Plot of Cell Voltage vs Time for multiple sampling rates of the SCD30

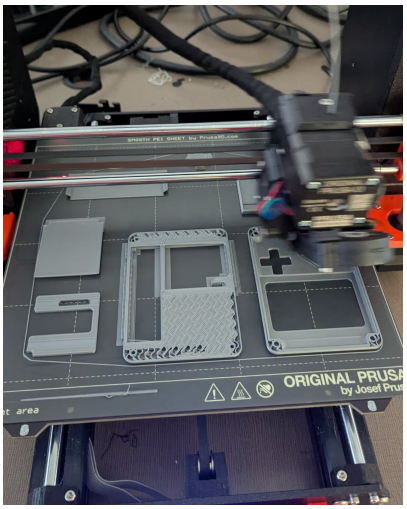
4.2 Sensor Accuracy

Although we did not have access to a fully controlled environment to test our prototype, performance was validated by comparing values to several thermostats around campus. Overall, the values reported for temperature, humidity, and CO_2 (the only values displayed on the thermostats) are very similar, which is expected given the high accuracy of the chosen sensors.

Figure 14: Comparison between Canary and various thermostats

4.3 Prototype Photos

Below are several pictures depicting various aspects of the prototype.

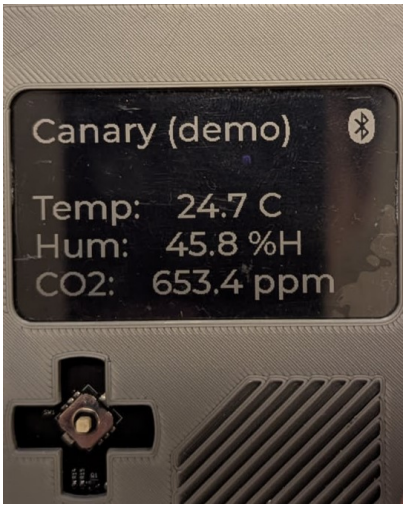


3D printing the case

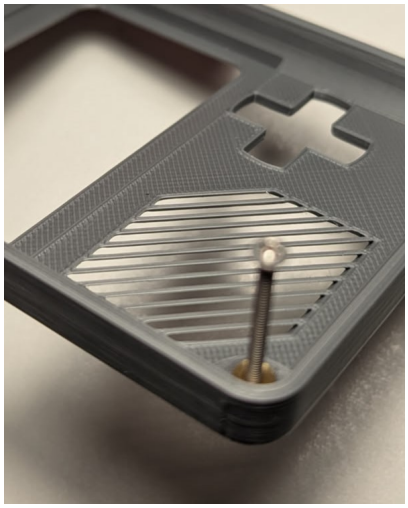
Front and back shells

PCB resting in case

CO_2 sensor installed



Sample display resolution



Sensor grille

Figure 15: Images depicting the construction of the Canary prototype

4.4 Application UI

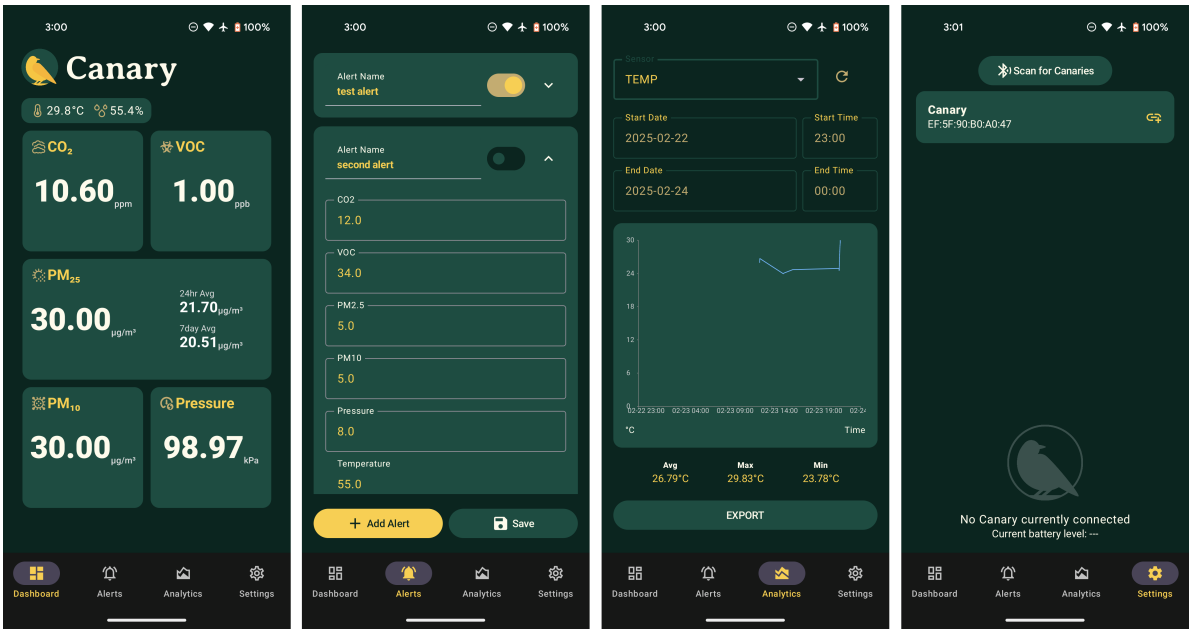


Figure 16: Screenshots of different tabs in the Canary mobile app

Screenshots of the application are shown in Figure 16.

When planning this project, we made accessibility an essential non-functional specification for the mobile app. People of any background, age, technological literacy, and physical ability should be able to use Canary, requiring both the hardware device and the mobile application to be accessible. As a result, the mobile application was created with W3C WCAG 2.0 guidelines in mind [6]. Font sizes are adjusted relative to the system default, allowing users who need larger fonts to benefit automatically. Additionally, the contrast ratio of text is kept above standard levels for maximum visibility. All pressable elements (e.g. buttons, AKA touch targets) are kept large, and almost all data are kept within the initial viewport, minimizing the need to scroll. Touch targets are also distinct and provide clear feedback (ripples, change in color) when pressed.

5 Discussion and Conclusions

5.1 Evaluation of the Final Design

The Detailed Design in section 3 and the Prototype Data in section 4 of the report provide clear evidence into the design and result of the project. The implementation of the final design was able to meet all numbered essential specifications. Section 3.1 describes the MCU selection and how the choice was able to meet specifications 5 and 6 where the Bluetooth low energy and data storage components were both achievable. For specifications 1 and 2 which were key requirements for the project, section 3.2 outlines the choices for the selection of the sensors and how the selected sensors meet the accuracy requirements.

We were highly confident that we would complete the project and achieve the essential design specifications for the symposium. The application design, initially identified as a potential risk, progressed smoothly and the necessary pages were implemented. With respect to the firmware, the design of the major subsystems (sensors, e-ink display, BLE, storage) has been validated and confirmed to be functional within the project specifications save for the power draw as discussed in section 4.1. The PCB design with the selected components was successfully completed and manufactured. The enclosure to house the device was successfully designed, printed and integrated with our device. Overall, the project specifications provided a strict guideline into the direction of the project and resulted in a device and application that met all the necessary requirements.

5.2 Use of Advanced Knowledge

This project used upper year ECE knowledge from ECE 356 (Databases), ECE 452 (Software Design) and ECE 463 (Power Converters). Our mobile app uses a local relational database to store data from the sensors. We use knowledge about indexes and normalization to make queries more efficient and save space. Software design is heavily used in both the mobile app, as well as the firmware, taking advantage of common software design principles such as SOLID and layered architecture. ECE 452's course project involves the development of an Android app, which gives us additional domain-specific skills to help build our own app. Power system efficiency is incredibly important for battery operated devices. Leveraging our knowledge from ECE 463, we chose the correct topology for each of our power converters and properly spec'ed their supporting components such as the output capacitor and main inductor. For battery life and charging described in section 3.3, the power system was designed to meet specifications 3 and 4 effectively and achieved that goal.

5.3 Creativity, Novelty, Elegance

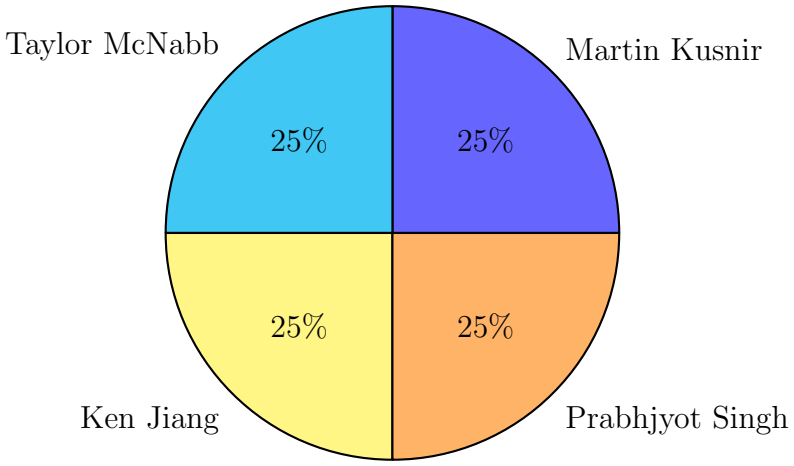
A novel aspect in this space is the standalone functionality of the device. Most air quality monitors require a mobile app to function in general. Canary comes equipped with a screen for headless functionality, and uses generic 18650 battery cells that can be hot swapped to overcome the diminished battery life. Additionally, the enclosure design includes consideration for the air intake to the sensors. Each of the sensors measuring ambient air are isolated from the PM2.5 sensor, which draws in air using a fan. This solution ensured isolation of each sensor, improving accuracy. The enclosure is also designed to be easily mountable, with 1/4"-20 (tripod) threads allowing for a wide variety of third-party mounting options. Additionally, the design of the historical analysis page enables the user to export all sensor data for certain date ranges which the user can then share with healthcare professionals.

The project is sufficiently complex, encompassing both hardware and software design. On the software side, it involved managing data structures, storing data, building a front-end app from scratch, and establishing a BLE connection between a mobile device and our hardware. Hardware challenges included designing a power system with the ability to be recharged and a battery that can be hot swapped for extended usage, creating a PCB layout to incorporate various components and sensors essential to the function of our project, and developing a sensor array with five sensors providing data to a micro-controller that outputs to both a display and a phone app.

5.4 Quality of Risk Assessment

The risk assessment done at the beginning of the project design cycle proved to be accurate descriptions of the problems we had during the development of the project. Section 4.1 provides a detailed account of the challenges the power subsystem presented in the project. While the subsystem's design itself did not cause any issues, the predicted sensitivity of the system's power consumption was confirmed by an unexpected manufacturing error, which significantly reduced battery life. Despite this setback, the research and design efforts ensured a still-usable battery life, along with a functional pass-through power option and charging capability. The app development mitigation discussed in the risk assessment ended up being a larger problem than anticipated and resulted in the platform change from React Native and Expo to Android Studio and Kotlin which resulted in smoother app development.

5.5 Student Workload



The project workload was evenly distributed across all group members. The unique skill sets of each member were taken into consideration when assigning tasks and designing each subsystem, allowing for effective division of the work.

6 References

[1] World Health Organization, *Ambient (outdoor) air pollution*, en, Dec. 2022. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/ambient-(outdoor)-air-quality-and-health.

[2] K. Shairsingh, G. Ruggeri, M. Krzyzanowski, *et al.*, "Who air quality database: Relevance, history and future developments," *Bulletin of the World Health Organization*, vol. 101, no. 12, pp. 800–807, Dec. 2023. DOI: 10.2471/blt.23.290188. [Online]. Available: https://doi.org/10.2471/blt.23.290188.

[3] World Health Organization, *Household air pollution*, en, Dec. 2023. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/household-air-pollution-and-health.

[4] National Institute of Environmental Health Sciences, *Air pollution and your health*, en. [Online]. Available: https://www.niehs.nih.gov/health/topics/agents/air-pollution.

[5] S. Sousan, S. Regmi, and Y. M. Park, "Laboratory evaluation of low-cost optical particle counters for environmental and occupational exposures," *Sensors*, vol. 21, no. 12, p. 4146, Jun. 2021. DOI: 10.3390/s21124146.

[6] W3C, *Mobile accessibility: How wcag 2.0 and other w3c/wai guidelines apply to mobile*, Feb. 2015. [Online]. Available: https://www.w3.org/TR/mobile-accessibility-mapping.